THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ROBUST IMAGE CLASSIFICATION BASED ON PIXEL IMPORTANCE

CHAEWAN CHUN
SPRING 2022

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computer Science and Mathematics
with honors in Computer Science

Reviewed and approved* by the following:

Jia Li
Professor of Statistics
Thesis Supervisor

Danfeng Zhang
Professor of Computer Science
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

ABSTRACT

Machine learning had a massive impact on various applications such as speech recognition, computer vision, natural language processing, and health. Machine learning aims to transform big data into actionable intelligence that is capable of performing tasks that need human intelligence. However, recent research works have revealed the vulnerabilities in machine learning algorithms. The machine learning model can be attacked by manipulated adversarial data. In order to overcome the machine learning models' vulnerabilities, recent work has been active in making learning models robust against adversarial inputs and evaluating the robustness of different machine learning models.

In the image recognition domain, adversarial images are designed by manipulating the input images small enough that external observers cannot visually distinguish the difference. If the machine learning model was attacked with adversarial images, the model performance could be affected depending on the models' vulnerabilities. In this paper, a new machine learning method is developed to increase robustness against adversarial input. The main idea is to use explanation models to identify unimportant pixels in the image and then modify the training images by blurring those pixels. This strategy is a defense against overfitting and adversarial attacks. This method is evaluated by multiple data sets and compared with other approaches.

Table of Contents

List of Figures **iii**

List of Tables **iv**

ACKNOWLEDGEMENTS **v**

1 Introduction **1**

2 Background **3**
2.1 Adversarial Attack 4
2.1.1 Goal of the Adversarial Attack 5
2.1.2 Distance Metrics 6
2.1.3 CNN basics 6
2.2 Generating Adversarial Examples 7
2.2.1 Fast Gradient Sign Method 7
2.2.2 NewtonFool Method 8
2.3 Defense Against Attacks 8

3 Methods **9**
3.1 Approach 10
3.2 Apparatus/ Materials 10
3.2.1 Blurring Process 10
3.2.2 Heatmap-based Blurring 11
3.3 Evaluation 11

4 Experiments and Results **14**
4.1 Experiments 15
4.1.1 Experimental Setup 15
4.1.2 Experiment Procedure 16
4.2 Results 17

5 Conclusions and Discussions **22**
5.1 Conclusions 23
5.2 Further Discussions 23

Bibliography **24**

List of Figures

1.1 Caption to go in list of figures 2

2.1 Caption to go in list of figures 4
2.2 Caption to go in list of figures 4
2.3 Caption to go in list of figures 5
2.4 Caption to go in list of figures 8

3.1 Caption to go in list of figures 10
3.2 Caption to go in list of figures 11
3.3 Caption to go in list of figures 12
3.4 Caption to go in list of figures 12
3.5 Caption to go in list of figures 13
3.6 Caption to go in list of figures 13

4.1 Caption to go in list of figures 16
4.2 Caption to go in list of figures 18
4.3 Caption to go in list of figures 18
4.4 Caption to go in list of figures 18
4.5 Caption to go in list of figures 19

List of Tables

4.1 Name of table for list of tables 19
4.2 Name of table for list of tables 20
4.3 Name of table for list of tables 20
4.4 Name of table for list of tables 20
4.5 Name of table for list of tables 21

ACKNOWLEDGEMENTS

Throughout the writing of this dissertation, I have received great assistance and support.

I would first like to thank my thesis advisor, Professor Jia Li, for her patience, care, immense knowledge, and continuous support of research. Her insightful feedback pushed me to sharpen my ideas and brought my output to a higher level.

I would like to acknowledge my honors advisor, Professor Danfeng Zhang, for the valuable guidance throughout my studies. I got many great opportunities to complete my studies successfully.

Chapter 1

Introduction

In recent years, machine learning algorithms have significantly impacted health care, computer vision, finance, speech recognition, natural language processing, and automotive systems. Machine learning systems have shown achievement in challenging classification problems in image classification and speech recognition, etc. Due to these accomplishments, machine learning systems are applied in safety-critical tasks, and these tasks require the result to be highly accurate, stable, and reliable [20][14]. For example, autonomous vehicles use deep convolutional neural networks (CNNs) to recognize road signs [20]. However, it would lead to a dangerous scenario if the vehicle did not stop as a consequence of the CNN model failing to recognize the "STOP" sign on the road.

Figure 1.1: The image on the left is an original image of a stop sign. The image on the right is an adversary image of the image on the left with a small perturbation added. (Image credit: Jang et al. [10])

Thus, recent works have emphasized security as a big concern in using machine learning algorithms as it can cause serious consequences. Many recent studies have shown that several machine learning models are vulnerable to adversarial examples - "input formed by applying small but intentionally worst-case perturbations to examples from the data set, such that the perturbed input results in the model outputting an incorrect answer with high confidence" [8]. In computer vision scenarios, an image that is manipulated by a small perturbation can lead the machine learning models to provide wrong prediction outputs. For example, in Figure 1.1, two images appear to be the same to external observers. However, the image on the right, an adversarial image, can force a trained machine learning model to classify incorrectly. As a result, the existence of adversarial examples has cautioned researchers against directly applying machine learning models in safety-critical tasks [20]. Evaluating the machine learning model's robustness has become an important topic to determine if the model will safely and effectively achieve its goals.

In this paper, the model is designed with images that emphasize the critical pixels in each image. In order to classify images with higher accuracy, pixels that have low importance for the classification process are blurred based on the heatmap, where a luminosity highlights essential regions on the image that corresponds to a region of interest [17]. For example, any pixels indicating an object bird will be considered important in a given image set of birds. In contrast, pixels irrelevant to a bird will be indicated as unimportant. This paper analyzes the classification results by training and testing the model with sets of original images, entirely blurred images, and blurred images based on heatmaps. In addition, the model is evaluated through testing the model with adversarial images.

Chapter 2

Background

This section describes the requisite background. Then, focusing on the general background of adversarial attacks and the goal of the adversarial attacks, essential backgrounds such as distance metrics and the neural network are also covered. The last section describes the defense against the attack, which is closely related to the adversarial attack.

2.1 Adversarial Attack

As machine learning models are more widely deployed, security and safety problems have become a rising topic. Studies on machine learning models have reported that many machine learning models are vulnerable to adversarial situations [1]. An attacker can develop adversarial settings on any input such as image, video, and speech that can change the decision of a classifier. Forcing a machine learning model to output erroneous outputs by adding a slight noise, we call this adversarial attack.

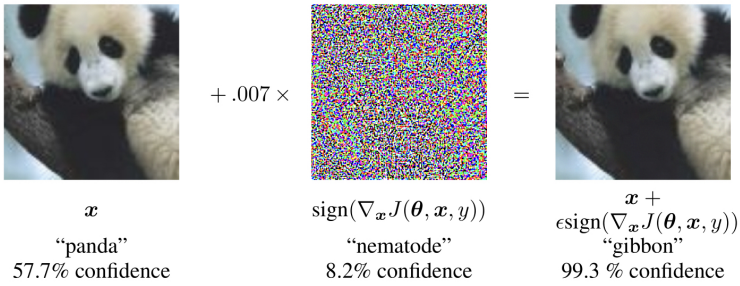

Figure 2.1: A fast adversarial example is applied to an image of a panda. By adding a small perturbation to the input, the classification of the image has changed. Here, elements in the small vector are equal to the sign of the elements of the cost function gradient. (Image credit: Goodfellow et al. [8])

Figure 2.2: The leftmost image is the original image of a horse. The other four are adversarial images with corresponding prediction labels, which are classified incorrectly. All five images appear very similar as the perturbations are very small (Image credit: Liu et al. [12])

White-box attack In this setting, an adversary has access to all the information of the target neural network [20]. Since the white-box attackers can access the model's architecture, parameters, gradients, etc., the adversary can craft adversarial examples with the full use of information [20].

Black-box attack In this setting, an adversary has no access to the inner configuration of DNN models [20]. Adversaries usually feed the input data and observe the output to read the

model's input-output relationship and target the weakness [20]. Black-box attacks are more practical than white-box attacks in applications since most model designers do not open source their parameters [20].

Gray-box attack In this setting, an attacker trains a generative model for producing adversarial examples in a white-box setting. Once the generative model is trained, the victim model is no longer needed by the attacker, and able to craft adversarial examples in a black-box setting [20].

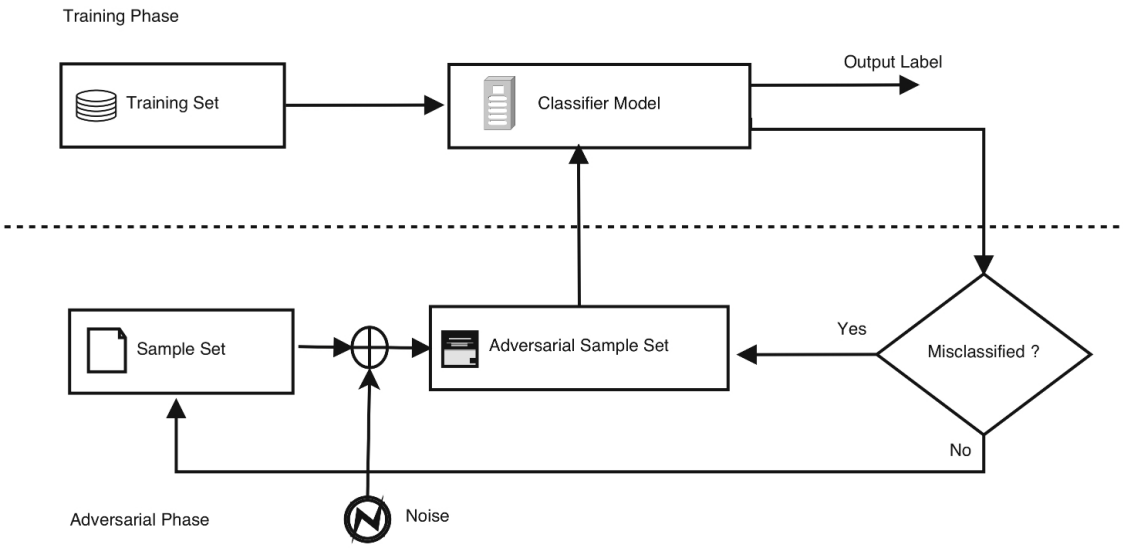

Figure 2.3: The basic architecture of Adversarial machine learning (Image credit: asha et al. [1])

2.1.1 Goal of the Adversarial Attack

At a high level, adversarial examples can be defined as inputs to machine learning models that an attacker intentionally designed to cause the model to make mistakes [20]. However, the precise goal of an adversarial attack can vary significantly based on the settings [2]. For example, adversarial attack's goal may be to cause misclassification on *any* input or *specific* input [2]. Adversarial attacks can be classified broadly as targeted attacks and non-targeted attacks, and poisoning attacks and evasion attacks with respect to the adversary's goal.

Targeted attack and non-targeted attack Given a valid input x and a target $t \neq C^*(x)$, a similar input x' such that $C(x') = t$ can be found while the distance between x and x' are close [3]. Distance Metrics will be discussed more in detail in the next section. An example x' in this property is called a "targeted" adversarial example. In non-targeted attack, we only search for an input x' so that $C(x') \neq C^*(x)$ and x,x' are close instead of classifying x as a given target class [3]. A non-targeted attacks are strictly less powerful than targeted attacks [3].

Poisoning attack and evasion attack Poisoning attacks are the attacking algorithms that allow an attacker to corrupt the training data by inserting several fake samples [1] [20][16]. This

type of attack is expected when the adversary has access to the training database [20]. For example, web-based repositories often collect malware examples for training, which makes the poisoning attack possible [20]. In an evasion attack, the attacker changes the behavior of machine learning models during the testing stage [1]. Classifiers and their parameters are fixed, but attackers craft some fraudulent samples so that classifiers cannot recognize them. For example, sticking a few pieces of stickers on the stop sign on the road can confuse the autonomous driving vehicles from recognizing the road sign correctly [20].

2.1.2 Distance Metrics

In order to measure the similarity, the use of a distance metric is necessary. There are some widely-used distance metrics in the literature for generating adversarial examples, which are L_p norms [4]. The L_p distance is defined as [3]:

$$\|x - x'\|_p = (\sum_{i=1}^{n} |(x - x')_i|^p)^{\frac{1}{p}}$$

Common distance metrics in detail:

- L_0-norm: L_0 distance measures the number of coordinates i such that $x_i \neq x'_i$, which corresponds to the number of pixels that have been altered in an image [3]. This method is useful when we want to limit the number of attack pixels [4].
- L_1-norm: L_1 distance measures the sum of the magnitudes of the distance between x and x'.
- L_2-norm: L_2 distance measures the standard Euclidean distance between x and x', which can remain small when there are many small changes to many pixels [3].
- L_∞-norm: L_∞ distance measures the maximum change to any of the coordinates: $\|x - x'\|_\infty = max(|x_1 - x'_1|, ..., |x_n - x'_n|)$. Using this measure, we can think of a maximum budget and any number of pixels can be modified up to this limit [3]. This methods minimizes the maximum element of the perturbation [4].

Distance metrics cannot be perfectly accurate in measuring human perceptual similarity. However, most existing work has picked these distance metrics, and we use L_1 norm in this paper on measuring the perturbation between original input and adversarial input [3]. Also, note that we report the distance metric on the range $[0, 1]$ as changing a pixel in a grey-scale image.

2.1.3 CNN basics

A Convolutional Neural Network (CNN) consists of an input layer, a number of hidden layers, and an output layer [12]. With some input, such as images, CNN propagates the data through a series of linear and non-linear operations, which we call neurons. Each hidden layer transform all the given input and outputs a classification prediction [12]. The CNN is trained by configuring the parameters of the filter in each layer [12].

Formally, a neural network is defined as function F such that $z = F(x)$ where $x \in \mathbb{R}^N$ and $z \in \mathbb{R}^m$. Here, z is an array $z = [z_1, z_2, ..., z_m]$, where z_i is the prediction probability of

class i with $i = 1, 2, ..., m$ for an m-class classifier. Note that z is subject to: $0 \leq z_i \leq 1$, and $z_1 + z_2 + ... + z_m = 1$. Then, the label y, which is the classification prediction of a given input x, takes the highest prediction probability from z. In other words, $y = label(x) = argmax_i z = argmax_i F(x)$ [12]. For a deep neural network classifier which has multiple layers of neurons with the last neuron being a softmax layer, the neural network can be expressed as [12]:

$$F = input\, layer \circ F_1 \circ F_1 \circ ... \circ F_{k-1} \circ F_k \circ softmax$$

where

$$F_i(x) = f_i(w_i x + b_i),\ i = 1, 2, ..., k$$

From the equation above, f_i is the activation function of layer F_i, w_i is the model weights, and b_i is the bias [12]. When the input form is images, input x is either grey-scale image with one channel or an RGB image with three channels (red, green, blue) which makes each channel of pixel x_i takes values from $[0, 255]$ [12].

2.2 Generating Adversarial Examples

Methods for generating adversarial examples can be divided into four different categories: *Gradient-based, Score-based, Decision-based,* and *Neural model generated attacks* [1]. This paper only focuses on the gradient-based attack, which uses a simple idea from the concepts involved in back-propagation. Among many efficient gradient-based attacks, we only discuss two of the white-box attacks, *Fast Gradient Sign Method (FGSM)* and *NewtonFool method*.

2.2.1 Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) was originally designed for attacking deep neural networks by maximizing certain loss function $\mathcal{L}(\mathbf{x}; \mathbf{w})$, subject to an upper bound on the perturbation [4][10]. For each pixel, the fast gradient sign method uses the gradient of the loss function to determine in which direction the pixel's intensity should be changed (whether it should be increased or decreased) to minimize the loss function, then it shifts all pixels simultaneously. [3][7].

This method works both in targeted and non-targeted settings, and control either L_1, L_2, or L_∞ norm of the adversarial perturbation [15]. In the targeted settings using L_∞-norm, the adversarial perturbation is generated by $-\eta \cdot sign(\nabla_x \mathcal{L}(\mathbf{x}; \mathbf{w}))$ where η is a positive value and chosen to be sufficiently small and so as to be undetectable and **w** is the target label [3] [4]. Also, $sign$ refers to the specific calculation for the L_∞-norm. Here, the attack transforms the input **x** to reduce the classifier's loss when classifying it as a label **w** [15].

Given an image $\mathbf{x}_0$, the FGSM creates an attack **x** by

$$\mathbf{x}_{adv} = \mathbf{x} - \eta \cdot sign(\nabla_x \mathcal{L}(\mathbf{x}; \mathbf{w}))$$

For the non-targeted case, the adversarial perturbation is generated by $\eta \cdot sign(\nabla_x \mathcal{L}(\mathbf{x}; C(\mathbf{x}))$ [15]. Here, the attack transforms the input **x** to increase the classifier's loss it continues to classify as label $C(\mathbf{x})$). The strength of the FGSM depends on the attack strength, η since the norm of input and the adversarial input grows linearly with η [15].

2.2.2 NewtonFool Method

NewtonFool belongs to an non-targeted attack category that tries to minimize the likelihood $F_y(\mathbf{x}$ of the correct output label $y = C(\mathbf{x})$ by manipulating the gradient [1].

As seen in Equation (7) from Figure 2.4: the step size δ is adaptable: the first term will dominate when $F_y(\mathbf{x}_{adv})$ approaches or falls below $1/K$; the second term will dominate when $F_y(\mathbf{x}_{adv})$ is larger than $1/K$ (K is the number of classes) which implies $C(\mathbf{x}_{adv}) = y$ [15]. The tuning parameter η determines how aggressively the gradient descent attempts to decrease the probability of class y [15].

Algorithm 6 NewtonFool attack

Input:
$\boldsymbol{x}$: Input to be adversarially perturbed
η: Strength of adversarial perturbations
$i_{\max}$: Maximum number of iterations

1: $y \leftarrow C(\boldsymbol{x}), \boldsymbol{x}_{\text{adv}} \leftarrow \boldsymbol{x}, i \leftarrow 0$
2: **while** $i < i_{\max}$ **do**
3: Compute

$$\begin{aligned} \delta &\leftarrow \min\left\{\eta \cdot \|\boldsymbol{x}\|_2 \cdot \|\nabla F_y(\boldsymbol{x}_{\text{adv}})\|, F_y(\boldsymbol{x}_{\text{adv}}) - 1/K\right\}, \\ \boldsymbol{d} &\leftarrow -\frac{\delta \cdot \nabla F_y(\boldsymbol{x}_{\text{adv}})}{\|\nabla F_y(\boldsymbol{x}_{\text{adv}})\|_2^2} \end{aligned} \tag{7}$$

4: $\boldsymbol{x}_{\text{adv}} \leftarrow \text{clip}(\boldsymbol{x}_{\text{adv}} + \boldsymbol{d}, x_{\min}, x_{\max})$
5: $i \leftarrow i + 1$
6: **end while**

Output:
Adversarial sample $\boldsymbol{x}_{\text{adv}}$.

Figure 2.4: NewtonFool attack's algorithm (Image credit: Nicolae et al. [15])

2.3 Defense Against Attacks

Over the last few years, the field of defense strategies against adversarial attacks has become an arising topic. In recent years, many defenses have been proposed. However, most of them could be broken with more powerful or adapted attacks [5]. Due to the many broken defenses, the field is currently in a state where it is hard to judge the value of a new defense [5]. As evaluating the robustness of a machine learning model becomes important, we study defenses to adversarial examples [2]. In summary, we can list three common reasons for the motivation to study this field [2]:

- To defend the system against an adversarial attack
- To test the worst-case robustness of machine learning algorithms
- To measure the progress of machine learning algorithms toward human-level abilities

Chapter 3

Methods

3.1 Approach

The topics of image blur analysis have attracted attention in past years [13].Blurring is a general image degradation that can unintentionally or intentionally happen. Unintentionally blurred images are usually caused by the low-quality camera, camera shaking, object movement, and out-of-focus [9]. Thus, the whole image is blurred as the blur was unintentional. People intentionally blur the background to highlight salient objects as well. In intentionally blurred images, the blur only occurs on the unimportant backgrounds and objects, which filter out the secondary information and catch attention to important objects [9]. With this beneficial intentional blurring effect on images, we propose that the model can classify images better when it is trained with intentionally blurred images.

Unintentionally blurred images are valuable for recording essential moments, so classifying these is another important task. Hence, we are interested in how the model classifies entirely blurred images based on the image set used for the training.

3.2 Apparatus/ Materials

3.2.1 Blurring Process

Gaussian filter method from the package *scipy* was used for the blurring process. After testing the blurring level by modifying the standard deviation for the Gaussian kernel, we used the value 4 to apply this blurring method effectively. As seen in Figure 3.1, the images get more blurry as the standard deviations of the Gaussian filter value get higher.

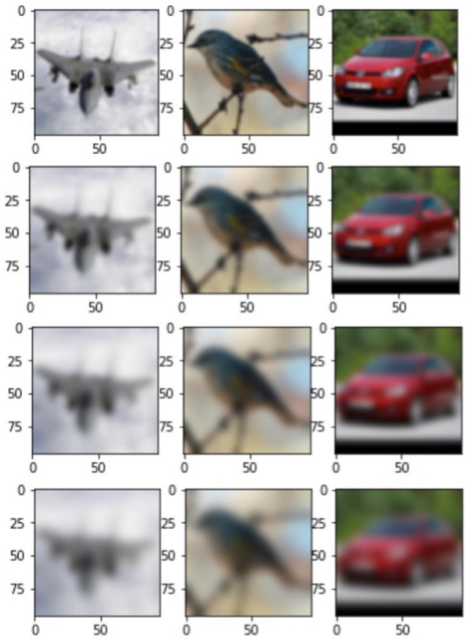

Figure 3.1: Images of an airplane, bird, and a car based on the blurring levels; From the top, blurred images with the standard deviation values 1, 2, 3, and 4 for Gaussian kernel

3.2.2 Heatmap-based Blurring

We use a heatmap to identify unimportant pixels to generate intentionally blurred images. The heatmap highlights the important patches which could make images more classifiable [11]. For our experiments, heatmap images were generated by four different methods: $Gradient$, $Input * Gradient$, $LRP - Epsilon$, and PGD. Given heatmap images, we select at least the top 15 percent important pixels of each image and blur the rest of the pixels (unimportant pixels). The top 15 percent important pixels were selected by identifying the highest 15 percent values on given heatmap images. Then, we keep the important pixels in their original form and replace unimportant pixels with pixels from entirely blurred images.

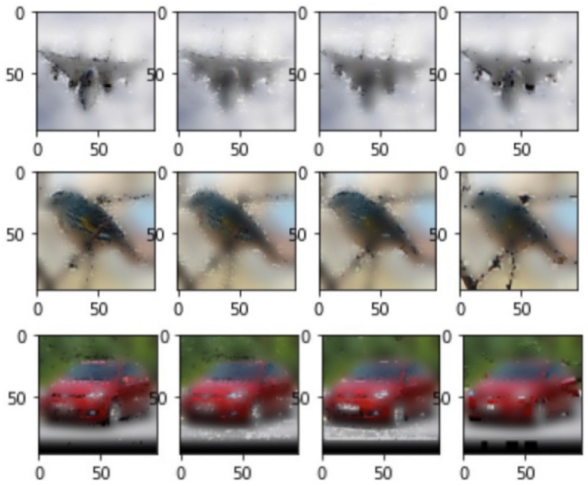

Figure 3.2: Blurred images of an airplane, bird, and a car based on the heatmap methods used. From the left, $Gradient$, $Input * Gradient$, $LRP - Epsilon$, and PGD methods were used to heatmap-base blurring

3.3 Evaluation

Evaluation of the proposed methods is mainly on the test accuracy of the model, which is the percentage of correct predictions for test data [21]. First, we train the model with original images and blurred images based on four different heatmaps as shown in Figure 3.3. Then, we test the model with original images, entirely blurred images, and blurred images based on four different heatmaps as shown in Figure 3.4. Finally, we evaluate the behavior of the machine learning model based on which train set and test set were used. To evaluate the robustness of this method, we use adversarial attack using Fast Gradient Sign Method and NewtonFool Method, as explained in Section 2.2. Generating adversarial attacked images can be in two ways 1)attack original images then blur attacked images based on heatmaps (Figure 3.5) or 2)blur original images based on heatmaps then attack (Figure 3.6). On the five different trained models, we test the model with attacked images based on whether we attack before or after heatmap-based blurring and which attacking methods are used.

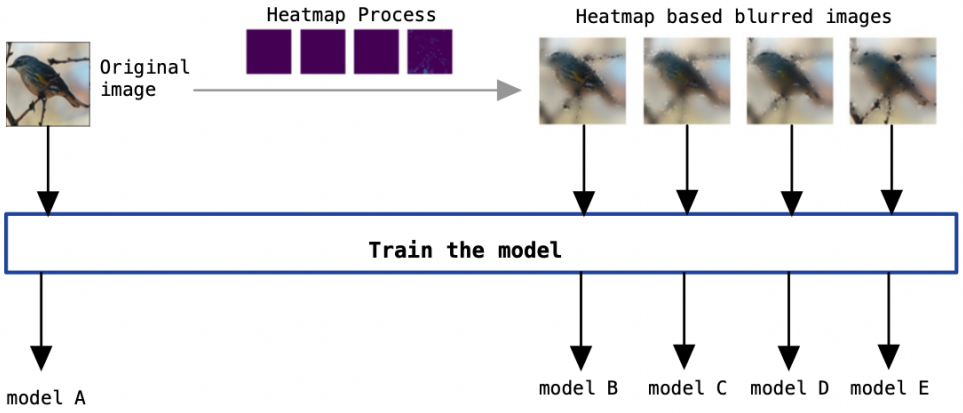

Figure 3.3: The CNN model is trained with original images and four different heatmap-based blurred images. This leads to five different trained models. Model A is when the model is trained with original images; models B, C, D, and E are trained with heatmap processed images using $Gradient$, $Input * Gradient$, $LRP - Epsilon$, and PGD respectively.

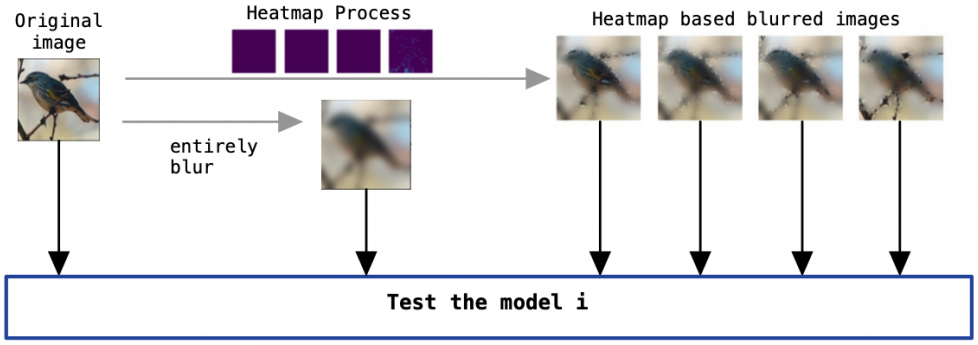

Figure 3.4: Five different trained models from Figure 3.3 is tested with original images, entirely blurred images, and four different heatmap-based blurred images. (i = A, B, C, D, E)

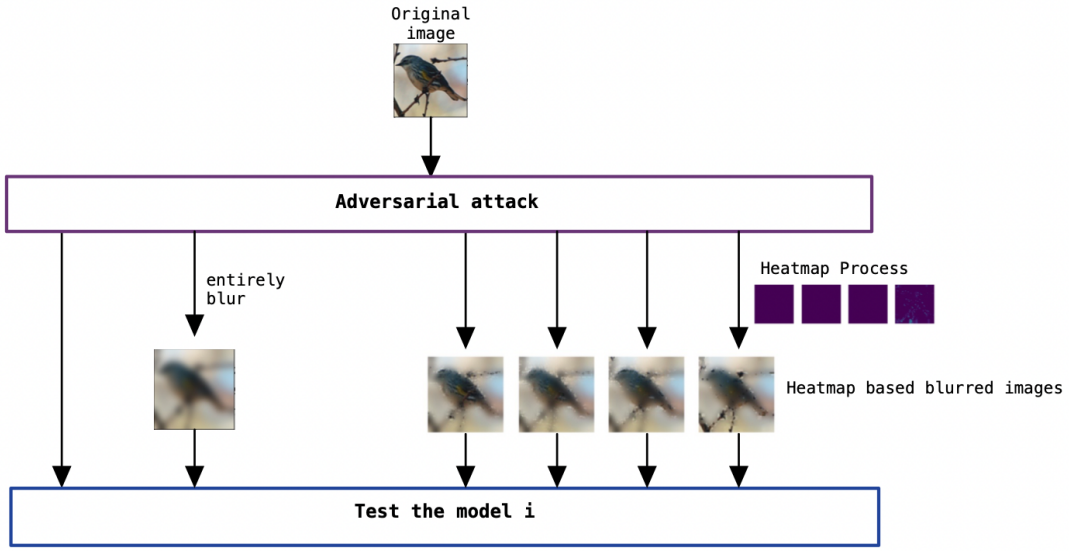

Figure 3.5: Original images are adversarial attacked, then gets entirely blurred or heatmap processed. Five different trained models from Figure 3.3 is then tested with those attacked images. (i = A, B, C, D, E)

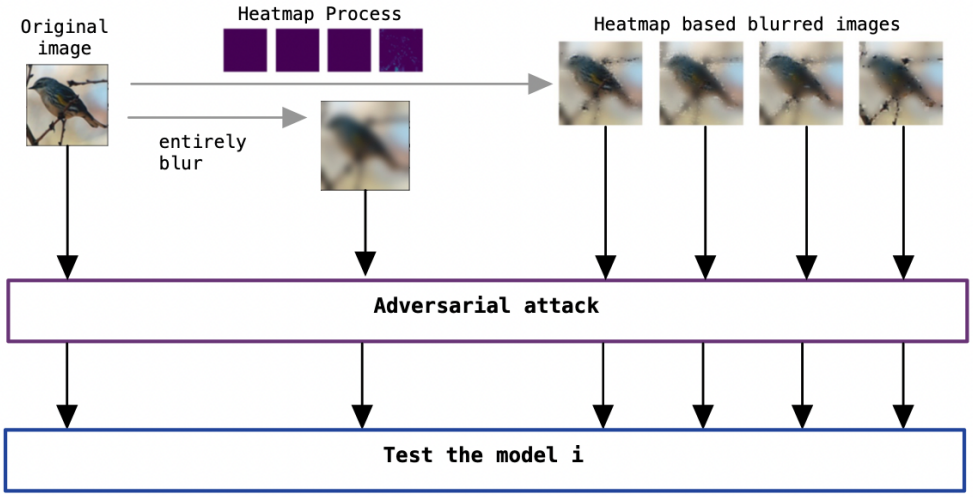

Figure 3.6: Original images, entirely blurred images, and heatmap-based blurred images are adversarial attacked. We test five different trained models from Figure 3.3 with those attacked images. (i = A, B, C, D, E)

Chapter 4

Experiments and Results

This section describes the setups for the experiment in detail and discusses the result. This experiment aims to examine 1) the effectiveness of the blurring method and 2) the robustness of the machine learning algorithm through an adversarial attack.

4.1 Experiments

4.1.1 Experimental Setup

Dataset

This experiment was performed with CIFAR-10, a dataset of 60,000 32x32 RGB images [19]. From this dataset, we used three classes where each class had 500 images, respectively. These three classes are Airplane, Bird, and Car. From 500 images in each class, we split into two subsets, the training set with 400 images and the test set with the rest of 100 images. In summary, 1200 images (400 images * 3 classes) are used for training, and 300 images (100 images * 3 classes) are used for testing.

We now have the original images and blurred images based on four different heatmaps. For convenience, along with this experiment, we call these five different image sets: original images, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, and images blurred by the PGD scheme.

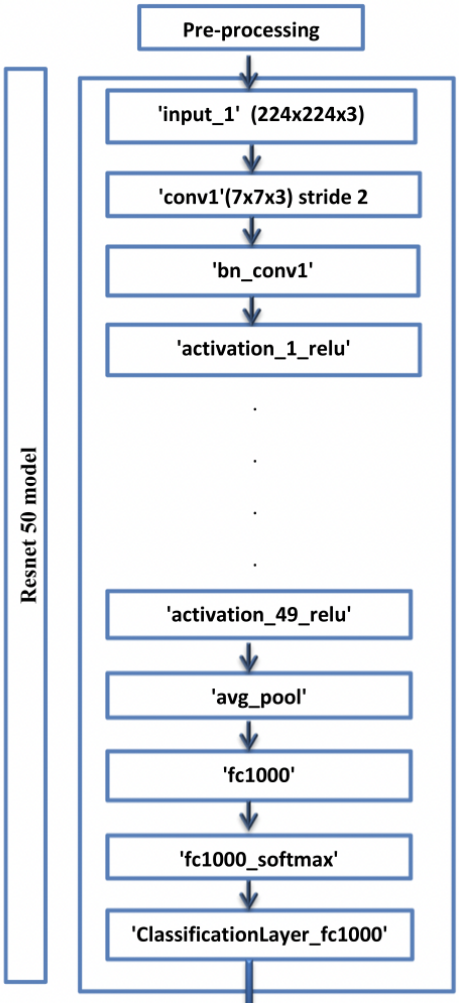

Figure 4.1: ResNet50 architecture (Image credit: Marwa et al. [6])

Model

We used the provided $keras$ ResNet50, a convolutional neural network (CNN) that is 50 layers deep. ResNet is known to be an excellent deep learning architecture due to its easy performance on optimization and its high accuracy [6]. The more detailed layer is shown in Figure 4.1.

4.1.2 Experiment Procedure

The model is trained with five different image sets: 1200 images from original images, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, or images blurred by the PGD scheme, respectively. On these five different trained models, we test the model using six different methods: the rest of 300 images from each original image, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, or images blurred by the PGD scheme, respectively, and the 300 images of entirely blurred original images.

Adversarial Attack

Adversarial attacking steps would be different whether we heatmap-based blur the images before the attack or after the attack.

Adversarial attacking steps for applying the heatmap-based blurring before the attack would be described as follows:

1. generate adversarial attacked images by attacking the six different test image sets (300 images from each original image, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, or images blurred by the PGD scheme, and the 300 images of entirely blurred original images)

2. test the model with the six different $attacked$ test image sets

Adversarial attacking steps for applying the heatmap-based blurring after the attack would be described as follows:

1. generate adversarial attacked images by attacking the original test image set (300 images)

2. entirely blur the $attacked$ original image set

3. apply heatmap-based blurring to the $attacked$ original image set using four different heatmap methods: $Gradient$, $Input * Gradient$, $LRP - Epsilon$, PGD

4. test the model with the six different $attacked$ test image sets

4.2 Results

The evaluation follows to determine the performance of the learned models. The four different test accuracy graph is generated based on which adversarial attack method was used and whether the images were attacked before or after the heatmap-based blurring. The graph x-axis represents what the model is trained with: original images, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, or images blurred by the PGD scheme, which corresponds to the model A, B, C, D, and E from the Figure 3.3. The y-axis shows the testing accuracy. Each bar shows what the model was tested with.

The first two graphs, Figure4.2 and Figure4.3, describes the work from Figure 3.5. These are the results when we apply heatmap-based blurring $after$ attacking the original images. The light-colored bar shows the testing accuracy on standard testing sets: original test images, entirely blurred images, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, or images blurred by the PGD scheme, while the line bars show the testing accuracy of the adversarial attacked images.

The last two graphs, Figure4.4 and Figure4.5, describes the work from Figure 3.6. These are the result when we apply heatmap-based blurring $before$ attacking images. The light-colored bar shows the testing accuracy on regular testing sets: original test images, entirely blurred images, images blurred by the Gradient scheme, images blurred by the Input*Gradient scheme, images blurred by the LRP epsilon scheme, or images blurred by the PGD scheme in order. The dark-colored bar shows the testing accuracy of the adversarial attacked images.

In next four graphs, 'original' bar (dark gray bar) corresponds to original images, 'entireBlur' bar (light gray bar) corresponds to entirely blurred images, 'Gradient' bar (red bar) corresponds to images blurred by the Gradient scheme, 'Input*Gradient' bar (yellow bar) corresponds to images blurred by the Input*Gradient scheme, 'LRP Epsilon' bar (green bar) corresponds to images blurred by the LRP epsilon scheme, and 'PGD' bar (blue bar) corresponds to images blurred by the PGD scheme. '_adv' on the end of the bar label describes the adversarial attacked images.

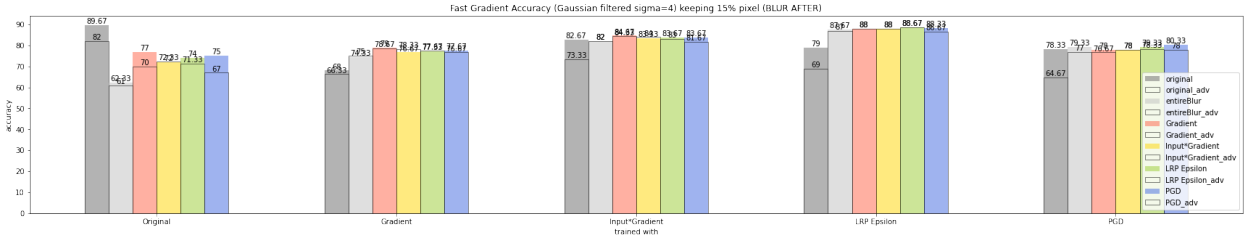

Figure 4.2: Test accuracy graph using FGSM for the adversarial attack, applying heatmap-based blurring after the attack

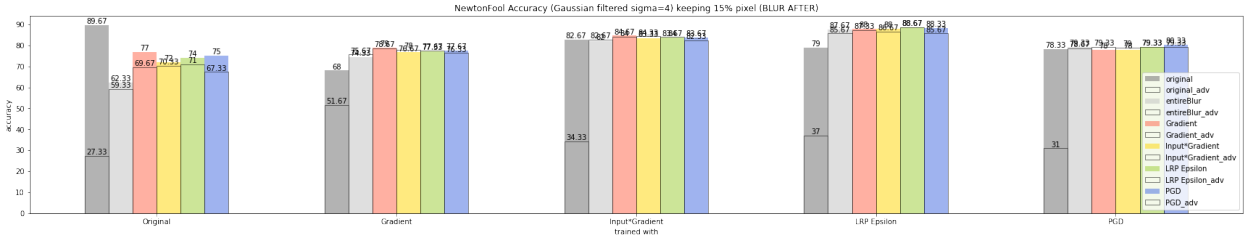

Figure 4.3: Test accuracy graph using NewtonFool Method for the adversarial attack, applying heatmap-based blurring after the attack

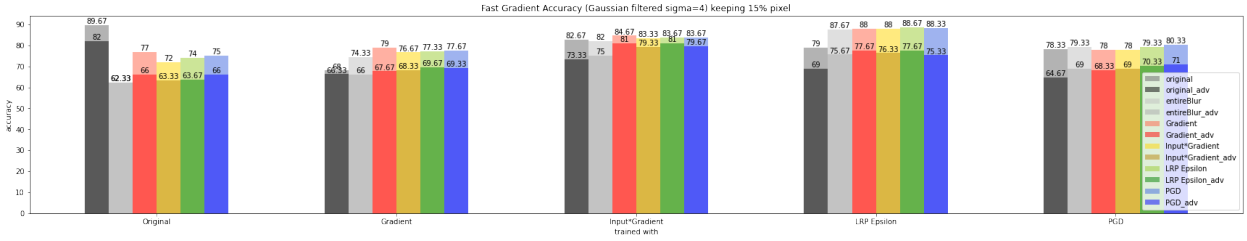

Figure 4.4: Test accuracy graph using FGSM for the adversarial attack, applying heatmap-based blurring before the attack

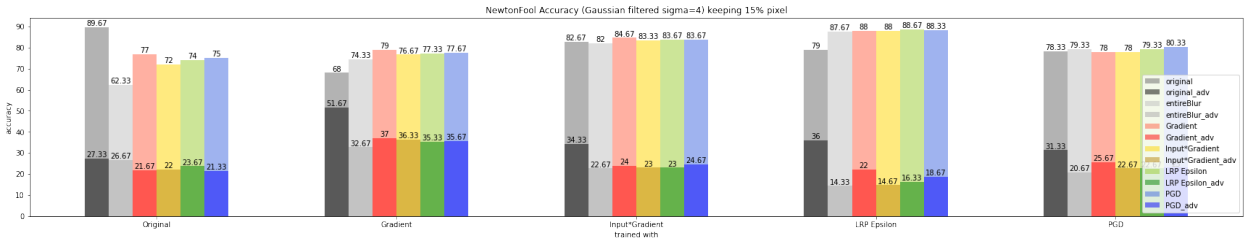

Figure 4.5: Test accuracy graph using NewtonFool Method for the adversarial attack, applying heatmap-based blurring before the attack

The following tables describe the test accuracy based on which image set was used for the training and testing. The first row describes the test accuracy when the model is tested with regular test image sets without attack. The rest describes the cases when the model is tested with attacked images. On the very left column, FGSM is when the images are heatmap-based blurred $before$ the attack using Fast Gradient Sign Method. FGSM$_A$ describes the image set that is heatmap-based blurred $after$ the attack using the Fast Gradient Sign Method. NewtonFool is when the images are heatmap-based blurred $before$ the attack using NewtonFool Method. NewtonFool$_A$ describes the image set that is heatmap-based blurred $after$ the attack using the NewtonFool Method. Each column describes how the images are processed: Blur is short for entirely blurred images, Gradient is short for images blurred by the Gradient scheme, In*Grad is short for images blurred by the Input*Gradient scheme, LRP-eps is short for images blurred by the LRP epsilon scheme, and PGD is short for images blurred by the PGD scheme.

Test Accuracy						
Original	Original	Blur	Gradient	In*Grad	LRP-eps	PGD
Original	89.67%	62.33%	77.00%	72.00%	74.00%	75.00%
FGSM	82.00%	62.33%	66.00%	63.33%	63.67%	66.00%
FGSM$_A$	82.00%	61.00%	70.00%	72.33%	71.33%	67.00%
NewtonFool	27.33%	26.67%	21.67%	22.00%	23.67%	21.33%
NewtonFool$_A$	27.33%	59.33%	69.67%	70.33%	71.00%	67.33%

Table 4.1: Test accuracy obtained by models trained with the original images

Test Accuracy						
Gradient	Original	Blur	Gradient	In*Grad	LRP-eps	PGD
Gradient	68.00%	74.33%	79.00%	76.67%	77.33%	77.67%
FGSM	66.33%	66.00%	67.67%	68.33%	69.67%	69.33%
FGSM_A	66.33%	75.00%	78.67%	78.33%	77.67%	76.67%
NewtonFool	51.67%	32.67%	37.00%	36.33%	35.33%	35.67%
NewtonFool_A	51.67%	75.67%	78.67%	78.00%	77.67%	76.33%

Table 4.2: Test accuracy obtained by models trained with the images blurred by the Gradient scheme

Test Accuracy						
In*Grad	Original	Blur	Gradient	In*Grad	LRP-eps	PGD
In*Grad	82.67%	82.00%	84.67%	83.33%	83.67%	83.67%
FGSM	73.33%	75.00%	81.00%	79.33%	81.00%	79.67%
FGSM_A	73.33%	82.00%	84.33%	84.00%	83.00%	81.67%
NewtonFool	34.33%	22.67%	24.00%	23.00%	23.00%	24.67%
NewtonFool_A	34.33%	82.67%	84.00%	84.33%	84.00%	82.33%

Table 4.3: Test accuracy obtained by models trained with the images blurred by the Input*Gradient scheme

Test Accuracy						
LRP-eps	Original	Blur	Gradient	In*Grad	LRP-eps	PGD
LRP-eps	79.00%	87.67%	88.00%	88.00%	88.67%	88.33%
FGSM	69.00%	75.67%	77.67%	76.33%	77.67%	75.33%
FGSM_A	69.00%	87.00%	88.00%	88.00%	88.67%	86.67%
NewtonFool	36.00%	14.33%	22.00%	14.67%	16.33%	18.67%
NewtonFool_A	37.00%	85.67%	87.33%	86.67%	88.67%	85.67%

Table 4.4: Test accuracy obtained by models trained with the images blurred by the LRP-epsilon scheme

Test Accuracy						
PGD	Original	Blur	Gradient	In*Grad	LRP-eps	PGD
PGD	78.33%	79.33%	78.00%	78.00%	79.33%	80.33%
FGSM	64.67%	69.00%	68.33%	69.00%	70.33%	71.00%
FGSM_A	64.67%	77.00%	76.67%	78.00%	78.33%	78.00%
NewtonFool	31.33%	20.67%	25.67%	22.67%	22.67%	23.00%
NewtonFool_A	31.00%	78.67%	79.33%	79.00%	79.33%	79.33%

Table 4.5: Test accuracy obtained by models trained with the images blurred by the PGD scheme

From Figure 4.2 to Figure 4.5, when the model is trained with blurred images based on heatmaps, the test accuracy was roughly the same whether the model is tested with original images or any blurred images. In contrast, when the model is trained with original images, the test accuracy drops when tested with entirely blurred images or blurred images based on heatmaps compared to when tested with original images. This result implies that the model trained with blurred images improves the model's robustness.

In Figures 4.2 and 4.3, when the model is trained with blurred images based on heatmaps, the test accuracy was roughly the same whether the model is tested with original images or any blurred images. Also, the result showed the test accuracy drops when the model is tested with attacked original images compared to the original images without attack. However, whether the test image set was attacked did not make much difference for any blurred images, including entirely blurred images and blurred images based on heatmaps.

Figures 4.4 and 4.5 show the general pattern of the test accuracy drop when the model is tested with attacked images. With any trained models, the test accuracy is lower when the model is tested with attacked images compared to non-attacked images in most bars. Especially when the NewtonFool Method is used for the adversarial attack, the accuracy dramatically drops by about 45%.

The interesting discovery is comparing the first two Figures, 4.2 and Figure 4.3, and the last two Figures, 4.4 and Figure 4.5. The accuracy does not drop when test images are attacked $before$ the heatmap-based blurring. However, the accuracy drops when test images are attacked $after$ the heatmap-based blurring. These different behaviors can conclude that an attacker knowing the heatmap-based blurring method before an attack can make the model more vulnerable.

Based on the result, we conclude 1) models trained using the blurred images based on heatmaps are more robust against attacks than those trained on the original images and 2) the testing accuracy stays about the same if the images are heatmap-based blurred after the adversarial attack; however, the testing accuracy drops if the images are heatmap-based blurred before the adversarial attack.

Chapter 5

Conclusions and Discussions

5.1 Conclusions

As machine learning systems became a significant part of our daily lives, security aspects of machine learning got increasingly important. Recent studies showed how many machine learning models are vulnerable to adversarial attacks. In this paper, we introduced the machine learning algorithm where we keep at least the top 15% important pixels and blur out the rest on each image based on the given heatmap images. We trained and tested the model with blurred images based on heatmaps compared with original images and also generated adversarial attacked images using the Fast Gradient Sign Method and NewtonFool Method to evaluate the robustness of the algorithm. The result addressed that blurred images based on heatmaps are more robust against attack than the original images. We also found that the testing accuracy does not get lower if the images are heatmap-based blurred after the attack.

5.2 Further Discussions

An evaluation of a machine learning model must allow a user to determine if the model will help safely and effectively achieve their goals; For this, evaluating performance on multiple, independently collected datasets is essential [18]. An experiment in this paper only used one dataset. Thus, testing this algorithm with more datasets will let us see if the result is consistent with a different dataset. Furthermore, the size of the dataset used in this experiment is small. Therefore, a more extensive dataset with more classes should be used to understand the introduced algorithm better.

It is an open problem to evaluate the robustness of the model against adversarial attacks. This paper used only gradient-based attacks; however, the result can look different if score-based, decision-based, or neural model-generated attacks were used. Moreover, using different attacking methods other than the Fast Gradient Sign Method or NewtonFool Method would be useful to see how the test accuracy behaves.

Bibliography

[1] S. Asha and P. Vinod. Evaluation of adversarial machine learning tools for securing ai systems. *Cluster computing*, 25(1):503–522, 2021.

[2] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness, 2019.

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.

[4] Stanley H. Chan. Ece 595ml: Machine learning i, Jan 2020.

[5] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, 2020.

[6] Marwa Elpeltagy and Hany Sallam. Automatic prediction of covid19 from chest images using modified resnet50. *Multimedia tools and applications*, 80(17):26451–26463, 2021.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[9] Rui Huang, Mingyuan Fan, Yan Xing, and Yaobin Zou. Image blur classification and unintentional blur removal. *IEEE access*, 7:106327–106335, 2019.

[10] Uyeong Jang, Xi Wu, and Somesh Jha. Objective metrics and gradient descent algorithms for adversarial examples in machine learning, 2017.

[11] Thi-Thu-Huong Le, Hyoeun Kang, and Howon Kim. Robust adversarial attack against explainable deep classification models based on adversarial images with different patch sizes and perturbation ratios. *IEEE Access*, 9:133049–133061, 2021.

[12] Kang Liu, Haoyu Yang, Yuzhe Ma, Benjamin Tan, Bei Yu, Evangeline F. Y. Young, Ramesh Karri, and Siddharth Garg. Adversarial perturbation attacks on ml-based cad: A case study on cnn-based lithographic hotspot detection, 2020.

[13] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[15] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M. Molloy, and Ben Edwards. Adversarial robustness toolbox v1.0.0, 2018.

[16] Izaskun Oregi, Javier Del Ser, Aritz Pérez, and José A. Lozano. Robust image classification against adversarial attacks using elastic similarity measures between edge count sequences. *Neural Networks*, 128:61–72, 2020.

[17] Konpat Preechakul, Sira Sriswasdi, Boonserm Kijsirikul, and Ekapol Chuangsuwanich. Improved image classification explainability with high-accuracy heatmaps. *iScience*, 25(3):103933–103933, 2022.

[18] Adarsh Subbaswamy, Roy Adams, and Suchi Saria. Evaluating model robustness and stability to dataset shift, 2020.

[19] Mason Swofford. Image completion on cifar-10, 2018.

[20] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review, 2019.

[21] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V. Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

CHAEWAN CHUN

EDUCATION

Pennsylvania State University Aug 2018 – May 2022
Schreyer Honors College
Bachelor of Science in **Computer Science**
Bachelor of Science in **Mathematics**
Minor: **Statistics**

AWARDS & GRANTS

President's Freshman Award, *Pennsylvania State University* Apr 2019
for academic achievement awarded by the chancellor of the Penn State

Evan Pugh Scholar Award, *Pennsylvania State University* Apr 2020
for academic achievement awarded by the chancellor of the Penn State

Student Engagement Network Innovation Grant, *Pennsylvania State University* Apr 2021
for research during summer 2021 with Prof. Jia Li

Leadership Scholarship, *Pennsylvania State University* May 2021
for leadership engagement in Women in Engineering program

Paul MacDonald Open Doors Scholarship, *Pennsylvania State University* Nov 2021
for leadership and academic achievement in College of Engineering

BP People's Choice Award, *Pennsylvania State University* Nov 2021
for Capstone Project Design achievement with Volvo Truck in College of Engineering

RESEARCH EXPERIENCE

Digital Sensing in Eye-related Applications Aug – Dec 2019
Abington College Undergraduate Research Activities (ACURA)

- Collaborated with Prof. Yi Yang in the development of hardware such as handheld measuring devices that assistant people with vision impairment
- Calculated the distance between the eyes and device along with the angle of the vision area with different sensors and analyzed the measurements accordingly to find the suitable sensor for the device

Explainable Image Classification Aug 2020 – May 2021
Pennsylvania State University

- Collaborated with Prof. Jia Li on image classification to predict the class of an image based on its distance to prototypes (representative images of a class)
- Created Prototypes for each class based on either CNN-based features or other classical image features such as edges and used the Wasserstein distance based on optimal transport

Robust Image Classification based on Pixel Importance May 2021 – Present
Pennsylvania State University

- Collaborated with Prof. Jia Li on improving the robustness of image classification algorithms by leveraging assessment of pixel importance
- Studied a training mechanism that exploits augmented images with pixels of low importance blurred and evaluated the approach by testing using images altered via multiple adversarial attack schemes

CHAEWAN CHUN

PROJECTS

Independent Studies on Discrete Mathematics Aug – Dec 2019
Pennsylvania State University
- Proposed and solved advanced problems in number and graph theory
- Discussed my results in weekly meetings with Prof. Michael Tepper

Computer Vision (CMPEN 454) Aug – Dec 2020
Pennsylvania State University
- Implemented forward and inverse camera projection
- Performed triangulation from two cameras for 3D construction
- Employed the four simple motion detection algorithms: simple background subtraction, simple frame differencing, adaptive background subtraction, and persistent frame differencing

Stochastic Modeling (STAT 416) Jan – May 2021
Pennsylvania State University
- Wrote a short paper on "Hidden Markov Models (HMM) in Part-of-Speech Tagging"
- Proposed the extension of the existing HMM

Numerical Solutions of Ordinary Differential Equations (CSE 551) Jan – May 2021
Pennsylvania State University
- Designed the experiment using different methods: Explicit Euler's method, Implicit Euler's method, Explicit Euler's method with trapezoidal adaptive step, 4^{th}-order Runge-Kutta method, Adams-Bashforth method
- Compared the accuracy and errors depending on the step size on each method

Nittany A.I. Challenge (Top 20 teams) Jan – Apr 2021
Pennsylvania State University
- Proposed an A.I. imagery and temperature screening robotic system for automated monitoring of body temperature and Covid-19 compliance for construction workers
- Analyzed the Information Retrieval – based data collected from the thermal camera to track worker body temperature, mask, and hard hat usage

Capstone Group with Volvo Truck Aug – Dec 2021
Pennsylvania State University
- Analyzed and cleaned the large dataset of size 181805x145 that is collected from customers' vehicles
- Designed the machine learning model by testing different algorithms such as Random Forests and Boosted Decision Trees (XGBoost) that learns the relationship between truck configurations and fuel efficiency with the highest testing accuracy
- Generated the three optimal recommendations and their truck configuration with importance features for the customers

Machine Learning and Algorithmic A.I. (CMPSC 448) (Honors) Aug – Dec 2021
Pennsylvania State University
- Used the non-linear classifiers: Boosted Decision Trees (XGBoost), Random Forests, Support Vector Machines (SVM) with Gaussian kernel for classification of given dataset
- Used K-fold cross-validation and computed the test errors after selecting the optimal hyperparameters
- Implemented the k-means++ algorithms and clustered the given dataset, then created plots to understand how objective changes with number of iterations for the chosen number of clusters

CHAEWAN CHUN

ACTIVITIES/ LEADERSHIP

Mathematics Peer Tutor Jan – Jul 2020, Jan – May 2021
Penn State Learning

- Tutored fellow undergraduate students in mathematics courses from introductory math courses to advanced calculus courses
- Customized study strategies for students to improve their academic performance and understanding
- Encouraged students of diverse backgrounds to develop independent studying habits in STEM

Facilitated Study Group Coordinator, Women in Engineering Program Jan – May 2021
Penn State, College of Engineering

- Guided a group of female students through Programming and Computation course materials
- Organized coursework lectures and created the study guide that covers challenging topics
- Conducted weekly meeting to review the course materials and discuss concepts

Volunteer – Penn State Peer Mentor Collective Feb 2021 – Present
Penn State, Electrical Engineering and Computer Science

- Volunteered to guide five paired students studying computer science and electrical engineering through regular one-on-one sessions
- Helped navigate challenges and recognized opportunities including the career advice

Volunteer – Engineering Orientation Network (EON) Leader/ Mentor Aug 2021 – Present
Penn State, College of Engineering

- Volunteered to welcome a group of incoming computer science students through the orientation activities: Dean's Speech, Q&A activities on engineering program, a design competition, and a campus tour
- Guided a dozen of assigned freshman year students to have them successfully adjust to the computer science program

Learning Assistant for Programming Language Concepts (CMPSC 461) Aug 2021 – Present
Penn State, College of Engineering

- Held office hours weekly to guide a class of 300+ students through the coursework material
- Monitored and managed class questions daily on the online platform Piazza
- Maintained close communication with students through emails

TECHNICAL SKILLS

- Programming Languages: Python, Octave, MATLAB, R, Java, C; Arduino, Quartus, Multisim, Verilog
- Deep Learning Frameworks: TensorFlow, Keras, PyTorch
- Programming Libraries: OpenCV, cuda, scikit, numpy, matplotlib, SciPy, pickle, pandas